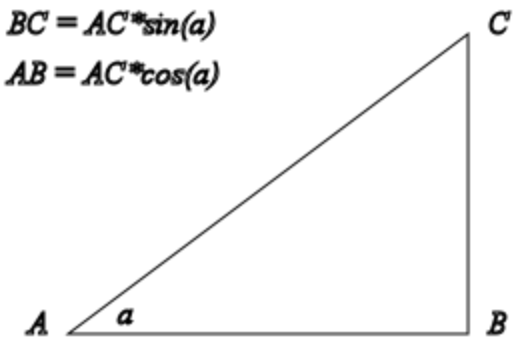
HTML5 CANVAS

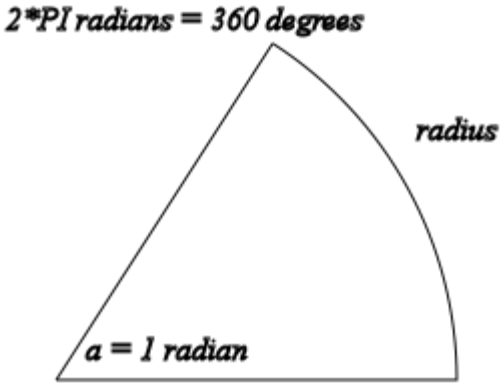
Canvas element - circles and ellipses

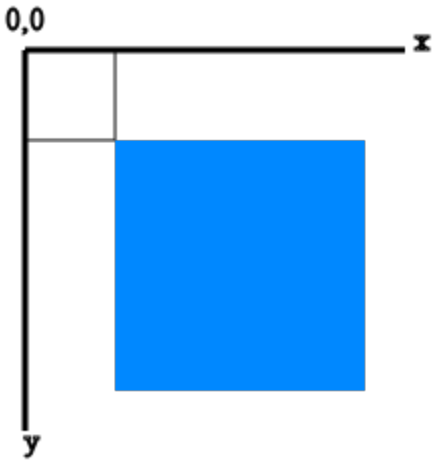
Several new elements have been introduced with the release of hTML5. Perhaps the most significant, is the canvas element which now makes it possible to draw, add text and create animations using Javascript. It is also possible to manipulate images. To make full use of the canvas element requires a knowledge of the Javascript **Math object**, and it's also helpful if you know how to convert between **rectangular** and **polar coordinates.** In common with most other program languages, angular coordinates are measured in **radians.** A quick maths refresher is shown below, and a more detailed explanation can also be found here

> Because the <canvas> element is still relatively new this will not work in older browsers (e.g. Internet Explorer versions below IE9). A means of providing alternate (fallback) content is therefore required for a browser doesn't support the canvas element.
>
> Of course, the ability to add drawings etc. to HTML isn't new. Similar features found in the canvas tag, are also available with Structured Vector Graphics (SVG).

Maths refresher







Here is a simple *helper* function to convert degrees to radians e.g.
var rad = radians(30);

```
function radians(degrees)
{
   //helper function to convert degrees to radians
   return parseInt(degrees)*Math.PI/180;
}
```

Example using HTML canvas element

The first example will draw an ellipse. For comparison purposes, the procedure used is similar to the example built using the Scratch programming language which was developed by MIT. The first task set up the canvas making it the same size as the Scratch stage. The two obvious program differences between Scratch and Javascript is a)the coordinates axis and b) the fact that angles are now measured in **radians**

The javascript program is show below. If you have built the Scratch examples then its instructive to compare the differences.

```
function drawCircle()
{
  // Create a circle or ellipse
  var canvas = document.getElementById("fig4");
  var cntx = canvas.getContext("2d");
  cntx.strokeStyle = "#000";            //set pen colour
  cntx.beginPath();                     //start new path
  //set values for major and minor axis
  var A = 200; var B = 120;
  var originx = 240; var originy = 180; //set origin
  cntx.moveTo(originx + A,originy);     //move to starting point
  //start withangle = 0 and loop round 2 PI radians
  //incrementing angle by 0.1 radian
  for(var angle=0;angle<2*Math.PI;angle+=0.1)
  {
    var x = A * Math.cos(angle) + originx
    var y = B * Math.sin(angle) + originy;
    cntx.lineTo(x,y);                  //draw line to new point
  }
  cntx.closePath();
  cntx.stroke();                       //create stroke
}
```

Using the canvas arc function

Here is an alternative way of drawing circles and ellipses by making use of the canvas arc function. The first function shown will clear the canvas, using the clearRect() method.

```
function clearCanvas()
{
  //clear the canvas
  var canvas = document.getElementById("fig4");
  var cntx = canvas.getContext("2d");
  cntx.clearRect(0, 0, canvas.width, canvas.height);
}

function drawArc()
{
  // Create a circle or ellipse
  var canvas = document.getElementById("fig4");
  var context = canvas.getContext("2d");
  context.save();
```

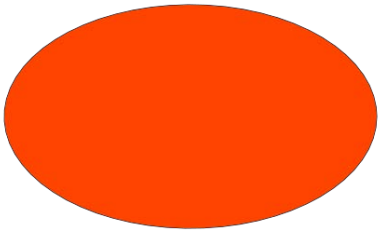
```
    var A = 120;                    //values for major/minor axis
    var sx = 1; var sy =1.25;       //set scale values
    var originx = canvas.width/2;
    var originy = canvas.height/2;  //set origin

    context.beginPath();            //start new path
    context.scale(sx,sy);
    context.strokeStyle = "#000";   //set pen colour
    // arc(originx,originy,radius,startAngle,endAngle, anticlockwise)
    // anticlockwise is either true or false

    context.arc(originx/sx,originy/sy,A,0,Math.PI*2);
    context.fillStyle = "#ff0";     //set pen colour
    context.fill();

    context.restore();
}
```

program output



Note: When calling the fill method, any open shapes will be closed automatically and it isn't necessary to use the closePath method.
Note: colors may be specified as "#rgb", "#rrggbb", rgb(rr,gg,bb) or rgba(rr,gg,bb,opacity). The rgba() function is similar to the rgb() function but it has one extra parameter. The last parameter specifies the transparency value. The valid range is between 0.0 (fully transparent) and 1.0 (fully opaque).

Source : http://www.soslug.org/node/1732